วารสาร มทร.อีสาน
ฉบับวิทยาศาสตร์และเทคโนโลยี
ISSN 1906-215X

การสร้างความปลอดภัยรหัสผ่านด้วยเทคนิคการแทรกซอลท์ร่วมกับแฮชฟังก์ชันโดยใช้อัลกอริทึม Bcrypt

An Improved Password Security Based on Salted Password and Hash Emotion using Bcrypt Algorithm

ปริญญา นาโท[1*] และศิรปัฐช์ บุญครอง[1]

Received: June, 2017; Accepted: August, 2017

บทคัดย่อ

การใช้งาน Cryptographic Hash ถือเป็นส่วนหนึ่งในวิธีการจัดเก็บรหัสผ่านที่มีความปลอดภัย แต่เนื่องจากแฮชฟังก์ชันนั้นมีความเร็วในการประมวลผล ซึ่งมีโอกาสที่จะถูกโจมตีด้วยเรนโบว์เทเบิลได้ ทำให้การจัดเก็บรหัสผ่านโดยผ่านกระบวนการแฮชฟังก์ชันเพียงอย่างเดียวไม่เพียงพอสำหรับการรักษาความลับของรหัสผ่านได้ ดังนั้นงานวิจัยนี้จึงได้ทำการนำเสนอวิธีการสร้างความปลอดภัยรหัสผ่านด้วยเทคนิคการแทรกซอลท์ร่วมกับแฮชฟังก์ชันโดยใช้อัลกอริทึม Bcrypt เพื่อให้รักษาความลับของรหัสผ่านก่อนทำการจัดเก็บลงในฐานข้อมูล ซึ่งตัวชี้วัดประสิทธิภาพด้านความปลอดภัยของรหัสผ่านทดสอบการโจมตีด้วยโปรแกรม Hashcat ผลจากการทดลองพบว่าการจัดเก็บรหัสผ่านโดยไม่ต้องใช้ซอลท์ด้วยอัลกอริทึม Bcrypt สามารถช่วยลดความเสี่ยงที่เกิดจากการโจมตีได้และถ้าหากใช้ซอลท์ควบคู่กัน ก็ทำให้ไม่สามารถทำการโจมตีด้วยโปรแกรม Hashcat ได้เช่นกัน

คำสำคัญ : ความปลอดภัยรหัสผ่าน; อัลกอริทึม Bcrypt; ซอลท์

[1] *Department of Information Technology, King Mongkut's University of Technology North Bangkok*
[*] *Corresponding Author E - mail Address: p.natho@hotmail.com*

Abstract

Cryptographic hash functions are ones of many methods used for password storing. However, cryptographic hash functions were designed to be very fast, which means there is a clack for Rainbow Table attack therefore, using hash functions alone is, therefore, not enough for storing passwords security. This paper proposed a method which uses a salt value with Bcrypt before storing the password in the database. The security was proved by using Hashcat. The results showed that Bcrypt on its own and Bcrypt with a salt value could withstand the attack.

Keywords: Password Security; Bcrypt Algorithm; Salt

บทนำ

ในปัจจุบันความก้าวหน้าทางด้านเทคโนโลยีการสื่อสาร การรับส่ง และการจัดเก็บข้อมูลในรูปแบบอิเล็กทรอนิกส์ถือว่ามีความสำคัญเป็นอย่างมาก เมื่อข้อมูลส่งจากผู้ส่งไปยังผู้รับจะต้องมีความปลอดภัย แนวความคิดหลักเกี่ยวกับความปลอดภัยของข้อมูล ได้แก่ การรักษาความลับของข้อมูล (Confidentiality) คือ ความสามารถในการรักษาความลับที่ไม่ให้ผู้ที่ไม่มีสิทธิ์สามารถอ่านข้อมูลได้ ความถูกต้องสมบูรณ์ของข้อมูล (Integrity) คือ ความสามารถในการรักษาความถูกต้องและสมบูรณ์ของข้อมูลว่าข้อมูลจะต้องไม่มีการสูญหายหรือถูกเปลี่ยนแปลงแก้ไขใด ๆ และความพร้อมใช้งาน (Availability) คือ ความสามารถในการตอบสนองเมื่อผู้ใช้งานต้องการใช้งานข้อมูลก็สามารถเรียกใช้ได้ทันทีตามสิทธิ์ที่ได้รับ [1]

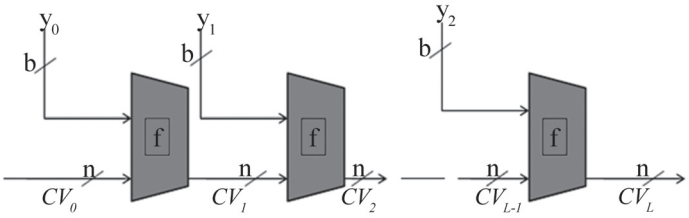
โดยแนวความคิดและจากวัตถุประสงค์ความปลอดภัยในด้านความพร้อมใช้งานข้อมูลนั้น จะมีเทคนิคที่ใช้สำหรับตรวจสอบการเข้าใช้ระบบงานต่าง ๆ คือ วิธีการยืนยันตัวตน (Authentication) สำหรับเข้าใช้งานซึ่งโดยส่วนใหญ่ของระบบการยืนยันตัวตนที่นิยมใช้นั้นจะประกอบด้วย การยืนยันตัวตนด้วยชื่อผู้ใช้ (Username) ร่วมกับรหัสผ่าน (Password) ซึ่งข้อมูลรหัสผ่านที่ใช้สำหรับยืนยันการเข้าระบบนั้นควรที่จะมีการจัดเก็บไว้ให้เป็นความลับ เนื่องจากถ้าหากผู้โจมตีหรือผู้ไม่ประสงค์ดีสามารถล่วงรู้ถึงรหัสผ่านได้ ข้อมูลที่จัดเก็บไว้เป็นความลับอาจจะไม่เป็นความลับอีกต่อไปได้ ซึ่งในการกำหนดรหัสผ่านถึงแม้จะมีกลไกที่สามารถเข้ามาควบคุมเพื่อช่วยทำให้การสร้างความปลอดภัยเพิ่มขึ้น เช่น การกำหนดความยาวและรูปแบบของรหัสผ่าน หรือแม้แต่การบังคับให้เปลี่ยนรหัสผ่านใหม่เมื่อครบระยะเวลาตามที่กำหนด [2] แต่ก็มีผู้ใช้งานจำนวนไม่น้อยที่กำหนดรหัสผ่านเดียวกันสำหรับการเข้าใช้งานในทุกระบบ เนื่องจากทำให้ผู้ใช้มีความง่ายต่อการจดจำในการเข้าใช้งาน ซึ่งถ้าหากรหัสผ่านที่จัดเก็บในระบบฐานข้อมูลไม่มีความปลอดภัยที่ดีแล้ว หากระบบถูกโจมตีอาจจะทำให้รหัสผ่านที่จัดเก็บไว้ในฐานข้อมูลถึงแม้ว่าบางครั้งการจัดเก็บรหัสผ่านจะมีวิธีการป้องกันโดยทำการนำรหัสผ่านที่จะจัดเก็บผ่านกระบวนการแฮชฟังก์ชันก่อนทำการจัดเก็บลงในฐานข้อมูล เช่น อัลกอริทึม MD5 ซึ่งอาจช่วยลดความเสี่ยงหรือโอกาสที่เกิดปัญหาในการถูกโจรกรรมข้อมูลได้ แต่เนื่องจากการประมวลผลของแฮชฟังก์ชันนั้นสามารถคำนวณผลลัพธ์ของค่าแฮชได้อย่างรวดเร็วซึ่งทำให้การโจมตีด้วยเรนโบว์เทเบิลเนื่องจากเรนโบว์เทเบิลเป็นตารางขนาดใหญ่

ที่สร้างขึ้นเพื่อทำการจัดเก็บค่าแฮชของรหัสผ่านเพื่อนำมาใช้เปรียบเทียบในการโจมตีก็สามารถทำงานได้รวดเร็วเช่นเดียวกัน [3] อีกทั้งค่าที่จัดเก็บในเรนโบว์เทเบิลที่สร้างมานั้นสามารถนำกลับมาใช้งานซ้ำได้อีก ทำให้รูปแบบการจัดเก็บรหัสผ่านโดยผ่านกระบวนการแฮชฟังก์ชันเพียงอย่างเดียวไม่เพียงพอสำหรับการรักษาความลับของรหัสผ่านได้ ดังนั้นงานวิจัยฉบับนี้จึงมีวัตถุประสงค์ในการนำเสนอวิธีการสร้างความปลอดภัยรหัสผ่านด้วยเทคนิคการแทรกซอลท์ร่วมกับแฮชฟังก์ชันโดยใช้อัลกอริทึม Bcrypt เนื่องจากอัลกอริทึม Bcrypt เป็นการเข้ารหัสตามอัลกอริทึม Blowfish และสามารถกำหนดการเพิ่มซอลท์อีกทั้งสามารถกำหนดจำนวนรอบในการทำซ้ำเพื่อให้การคำนวณหาค่าผลลัพธ์ของค่าแฮชมีการทำงานที่ช้าลง ซึ่งจะส่งผลให้การโจมตีทำงานได้ช้าลงด้วยเช่นเดียวกัน

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

1. แฮชฟังก์ชัน (Hash Function)

แฮชฟังก์ชัน [4] เป็นการแปลงรูปแบบของข้อมูลที่รับเข้ามาไม่ว่าข้อมูลต้นฉบับจะมีขนาดเท่าใดก็ตามจะถูกย่อยให้อยู่ในรูปแบบที่มีขนาดคงที่ ดังนั้นจึงไม่สามารถที่จะทำกระบวนการย้อนกลับเพื่อเรียกดูข้อมูลต้นฉบับได้ จะทำได้เพียงแค่ตรวจสอบว่าข้อมูลที่ส่งมานั้นถูกเปลี่ยนแปลงแก้ไขหรือมีการสูญหายของข้อมูลหรือไม่ โดยการทำงานของแฮชฟังก์ชันแสดงดังรูปที่ 1 ทั้งนี้ขนาดของค่าแฮชจะขึ้นอยู่กับอัลกอริทึมที่ใช้ เช่น อัลกอริทึม MD5 จะมีขนาดค่าแฮช 128 บิต

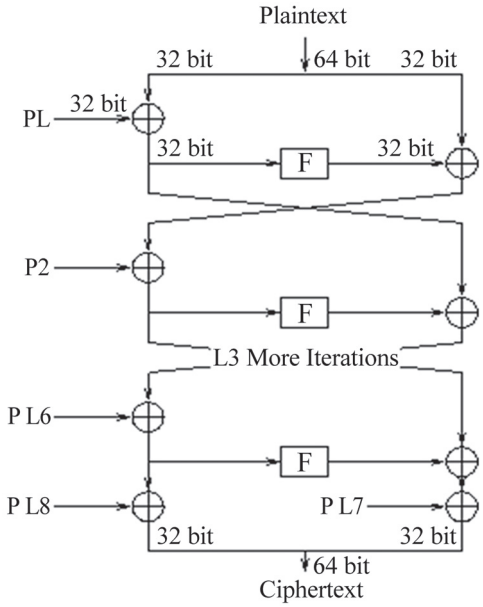


รูปที่ 1 การทำงานของแฮชฟังก์ชัน

แฮชเป็นฟังก์ชันที่ดีจะต้องมีคุณสมบัติของการกระจายที่ดี กล่าวคือ ข้อมูลเดียวกันเมื่อผ่านแฮชฟังก์ชันแล้ว จะต้องได้ผลลัพธ์เหมือนเดิมเสมอ และหากข้อมูลต่างกันเพียงเล็กน้อยเมื่อผ่านแฮชฟังก์ชันแล้วผลลัพธ์ที่ได้ควรมีค่าแตกต่างกันมาก สำหรับกระบวนการแฮชฟังก์ชันนั้น จะมีรูปแบบต่างกันตามอัลกอริทึม สามารถแบ่งเป็นกลุ่มหลัก ได้แก่ กลุ่มของ MDC (Modification Detection Codes) และกลุ่มของ MAC (Message Authentication Codes)

2. อัลกอริทึม Bcrypt

อัลกอริทึม Bcrypt [5] เป็นการเข้ารหัสตามอัลกอริทึม Blowfish และสามารถกำหนดการเพิ่มซอลท์อีกทั้งยังสามารถปรับเพิ่มขนาดจำนวนรอบการวนซ้ำเพิ่มขึ้น ซึ่งโดยปกติจะทำการวนซ้ำจำนวน 4 รอบ ทำให้การทำงานช้าลงได้ ดังนั้นจึงมีความสามารถในการป้องกันการโจมตีแบบ Brute Force ที่ใช้การค้นหารหัสผ่านที่ถูกต้องได้แม้ว่าจะมีความสามารถในการคำนวณเพิ่มขึ้นก็ตาม โดยการทำงานของอัลกอริทึม Bcrypt แสดงในรูปที่ 2



รูปที่ 2 การทำงานของอัลกอริทึม Bcrypt

อัลกอริทึม Bcrypt ในค่าแฮชข้อความหรือในไฟล์ข้อมูลที่จะทำการปกปิดรหัสผ่านจะมีค่าเริ่มต้นที่ประกอบด้วย $2a$ หรือ $2y$ เพื่อเป็นตัวระบุว่าเป็นรูปแบบการแฮชด้วยอัลกอริทึม Bcrypt ส่วนที่เหลือของแฮชข้อความรวมถึงค่าพารามิเตอร์ต่าง ๆ ประกอบด้วย ค่าซอลท์ 128 บิต และค่าแฮช 184 บิต [5]

3. **Salting**

Salt เป็นวิธีการหนึ่งที่สามารถเพิ่มความยาวให้กับรหัสผ่าน ค่าของ Salt อาจเป็นตัวอักษร ตัวเลขหรือตัวอักขระพิเศษที่ได้จากการสุ่มหรือกำหนดขึ้นมาเพื่อนำมาประกอบรวมกับรหัสผ่านเพื่อทำให้ค่าแฮชของรหัสผ่านที่รวมกับค่าซอลท์ [6] ยกตัวอย่างเช่น ผู้ใช้กำหนดรหัสผ่าน “password” หลังจากนั้นระบบทำการสุ่มค่าซอลท์ขึ้นมาค่าหนึ่ง สมมติได้ค่าเป็น “fh90$$PA28” เมื่อสุ่มค่าซอลท์ได้แล้วหลังจากนั้นจะทำการคำนวณค่าแฮชของรหัสผ่าน “password+Salt” แทนค่าได้ Hash(passwordfh90$$PA28) เมื่อผ่านกระบวนการแฮชฟังก์ชันด้วยอัลกอริทึม MD5 แล้วจะทำให้ผลลัพธ์ของค่าแฮช คือ 29e20a65f0d03364b0391174ac74b3 และเนื่องจากค่าซอลท์ของผู้ใช้แต่ละคนจะมีค่าที่แตกต่างกันเพื่อนำมารวมกับรหัสผ่านทำให้มีการคำนวณค่าผสมระหว่างรหัสผ่านกับค่าซอลท์นั้นมีความเป็นไปได้ของจำนวนที่เพิ่มมากขึ้นซึ่งส่งผลทำให้มีความยากต่อการโจมตีมากยิ่งขึ้นด้วย [7]

4. **การจัดเก็บรหัสผ่านและงานวิจัยที่เกี่ยวข้อง**

รูปแบบสำหรับการจัดเก็บรหัสผ่านนั้นมีเทคนิคและวิธีการจัดเก็บได้หลากหลายวิธี รวมถึงมีงานวิจัยได้นำเสนอเทคนิคและวิธีการต่าง ๆ ประกอบด้วยดังนี้

1) การจัดเก็บรหัสผ่านในรูปแบบข้อความปกติ (Plaintext)

การจัดเก็บวิธีการนี้เป็นวิธีการจัดเก็บรหัสผ่านที่ง่ายที่สุด โดยจะไม่มีการเข้ารหัสรหัสผ่านใด ๆ [8] ถึงแม้ว่าการจัดเก็บที่ดีควรจะเป็นรหัสผ่านที่มีความยาวอย่างน้อย 8 ตัวอักษร และต้องประกอบด้วย

ตัวอักขระพิเศษและตัวเลขผสมกับตัวอักษรก็ตาม [9] แต่ก็ยังทำให้การจัดเก็บรหัสผ่านแบบนี้มีความปลอดภัยน้อยมากเนื่องจากถ้าผู้โจมตีสามารถเข้าถึงฐานข้อมูลที่จัดเก็บรหัสผ่านได้สามารถที่จะมองเห็นรหัสผ่านนั้นได้ทันที

2) การจัดเก็บรหัสผ่านแบบผ่านแฮชฟังก์ชัน (Hash Password)

การจัดเก็บวิธีนี้จะทำการนำรหัสผ่านไปผ่านกระบวนการแฮชฟังก์ชันก่อน เช่น อัลกอริทึม MD5 อัลกอริทึม SHA1 หรืออัลกอริทึม SHA2 แล้วจึงนำผลลัพธ์ที่ได้ไปจัดเก็บลงในฐานข้อมูล [8], [10] ซึ่งการจัดเก็บรูปแบบนี้เมื่อผู้โจมตีสามารถเข้าถึงฐานข้อมูลที่จัดเก็บรหัสผ่านจะสามารถมองเห็นแค่เพียงค่าแฮชเท่านั้น โดยการจัดเก็บวิธีนี้สามารถถูกโจมตีได้โดยการใช้เรนโบว์เทเบิล ซึ่งเป็นตารางขนาดใหญ่ที่สร้างขึ้นเพื่อทำการจัดเก็บค่าแฮชของรหัสผ่านและค่าแฮชของรหัสผ่านนี้จะถูกนำมาใช้เพื่อการเปรียบเทียบกับเวลาที่นำไปใช้ในการโจมตี เนื่องจากค่าแฮชฟังก์ชันมีคุณสมบัติในการคำนวณทิศทางเดียวไม่สามารถทำการคำนวณย้อนกลับเพื่อหาผลลัพธ์ได้ ถึงแม้ว่าเรนโบว์เทเบิลที่สร้างมานั้นจะใช้พื้นที่ในการจัดเก็บค่อนข้างมาก แต่เรนโบว์เทเบิลที่ถูกสร้างมาแล้วนั้นสามารถถูกนำมาใช้งานซ้ำอีกได้ [10]

3) การจัดเก็บรหัสผ่านแบบผ่านแฮชฟังก์ชันสองครั้ง (Double Hash Password)

การจัดเก็บวิธีนี้จะเหมือนกับวิธีการจัดเก็บรหัสผ่านแบบแฮชฟังก์ชัน แต่จะแตกต่างกันคือวิธีการนี้จะนำค่าแฮชที่ได้จากการทำรอบที่หนึ่งมาทำการแฮชซ้ำอีกครั้ง ซึ่งการทำซ้ำนี้ผลลัพธ์ของค่าแฮชในรอบที่สองอาจเปลี่ยนแปลงหรือเหมือนกับรอบแรกก็ได้ [11] ซึ่งการจัดเก็บแบบนี้ผู้โจมตีสามารถเข้าถึงฐานข้อมูลที่จัดเก็บรหัสผ่านได้ก็จะมองเห็นเพียงค่าแฮชเท่านั้น ซึ่งมีความปลอดภัยมากกว่าการจัดเก็บรหัสผ่านแบบแฮชรอบเดียว แต่เนื่องจากความเร็วในการประมวลของค่าแฮชก็อาจทำให้ผู้โจมตีก็สามารถทำการทดสอบหาค่าของจำนวนรอบที่ใช้ในการแฮชแล้วสร้างเรนโบว์เทเบิลขึ้นมาใหม่เพื่อทำการโจมตีได้เช่นกัน [12]

4) การจัดเก็บแบบเติมซอลท์ในแฮชฟังก์ชัน (Salted Hash Password)

การจัดเก็บวิธีนี้จะทำการเติมค่าซอลท์ลงไปในรหัสผ่านก่อนแล้วค่อยนำไปผ่านกระบวนการแฮชฟังก์ชัน [13] โดยค่าซอลท์ คือ ค่าที่ถูกสร้างขึ้นมาอาจจะสร้างขึ้นมาโดยการกำหนดค่าแบบ prefix salt หรือแบบ suffix salt หรือแม้แต่การใช้เทคนิคการปรับค่าซอลท์ที่เหมาะสม [11] โดยแทรกซอลท์ลงในรหัสผ่านนั้นทำให้ผลลัพธ์ที่ได้มีความแตกต่างกันทำให้มีส่วนช่วยเพิ่มความปลอดภัยให้กับรหัสผ่านมากขึ้นทำให้การโจมตีด้วยตารางเรนโบว์เทเบิลทำได้ยากขึ้น แต่ถึงอย่างไรก็ตามเนื่องจากค่าซอลท์ที่ถูกสร้างขึ้นมาจะต้องทำการจัดเก็บลงในฐานข้อมูลด้วย ทำให้ผู้โจมตีทราบค่าซอลท์นั้นประกอบกับความเร็วในการประมวลผลของค่าแฮชจะสามารถทำการโจมตีได้เช่นกัน

วิธีดำเนินการวิจัย

งานวิจัยฉบับนี้มีวัตถุประสงค์เพื่อทำการสร้างความปลอดภัยรหัสผ่านด้วยการแทรกซอลท์ร่วมกับแฮชฟังก์ชันโดยใช้อัลกอริทึม Bcrypt โดยทำการพัฒนาด้วยภาษา PHP Version 7 ที่ทำงานบนความเร็ว CPU Core i3-3227U 1.90 GHz หน่วยความจำ 4 GB โดยเริ่มต้นการทำงาน เมื่อรับรหัสผ่านตั้งต้นเข้ามาแล้วระบบจะทำการสุ่มค่าซอลท์ที่ได้จากชุดข้อมูลดังตารางที่ 1 ซึ่งชุดข้อมูลที่นำมาสร้างซอลท์จะเป็นชุดข้อมูลตัวอักษร ชุดข้อมูลตัวเลขหรือชุดข้อมูลอักขระพิเศษ

ตารางที่ 1 ชุดตัวอักษรที่สร้างซอลท์

ชุดข้อมูล	ตัวอักษร
อักษร	abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ
ตัวเลข	0123456789
อักขระพิเศษ	!@#$%^&*()_+`~[]{}\ | ;:/?.><,'"

หลังจากนั้นจะทำการนำค่าซอลท์ที่ได้จากการสุ่มนำมาแทรกต่อท้ายของรหัสผ่านที่รับเข้ามาตั้งต้นผ่านกระบวนการแฮชฟังก์ชันโดยใช้อัลกอริทึม Bcrypt ยกตัวอย่างเช่น รหัสผ่านตั้งต้นคือ "password" ระบบทำการสุ่มค่าซอลท์ขึ้นมาจำนวน 8 ตัว ซึ่งการสุ่มแต่ละครั้งจะได้ค่าซอลท์ที่แตกต่างกัน ยกตัวอย่างเช่น ค่าซอลท์ที่ได้จากการสุ่ม DM<9^C%I ในการสุ่มค่าซอลท์ขึ้นมานี้เพื่อทำให้รหัสผ่านนั้นมีรูปแบบของรหัสผ่านที่ดี คือ มีความยาวอย่างน้อย 8 ตัวอักษร ผสมรวมกับตัวอักขระพิเศษหรือตัวเลข [9] หลังจากนั้นจะทำการนำค่าซอลท์ที่สุ่มได้แทรกรวมกับรหัสผ่านโดยวิธีการนำค่าซอลท์ต่อท้ายรหัสผ่านจะได้ "passwordDM<9^C%I" หลังจากนั้นจะนำค่าของรหัสผ่านที่รวมกับค่าซอลท์ที่สุ่มขึ้นมาทำการคำนวณหาค่าผลลัพธ์ของรหัสผ่านด้วยอัลกอริทึม Bcrypt จะทำให้ได้ผลลัพธ์ที่ได้จากการสุ่มค่าซอลท์ของรหัสผ่านมีขนาด 128 บิต ได้ค่า RlSj6p07xTval8BkDBKG/O และผลลัพธ์ค่าแฮชของรหัสผ่านมีขนาดความยาว 184 บิต ได้ค่า 2cN1.kgYk3kqUb3v5en13YLupn07.i2 โดยรูปแบบของผลลัพธ์เมื่อผ่านการทำงานแล้วจะประกอบด้วยส่วนตั้งต้นของอัลกอริทึม Bcrypt คือ $2y$ ถัดไป คือ ส่วนขยายจำนวนรอบของอัลกอริทึม เช่น 10 จะได้ค่า 10$ หลังจากนั้นจะต่อด้วยผลลัพธ์ค่าซอลท์ของรหัสผ่านที่ได้จากการสุ่ม และสุดท้ายจะต่อท้ายด้วยผลลัพธ์ค่าแฮชของรหัสผ่าน เมื่อนำมารวมกันแล้วแสดงได้ดังรูปที่ 3 ซึ่งผลลัพธ์ของค่าซอลท์ที่ได้จากการสุ่มของรหัสผ่านและผลลัพธ์ค่าแฮชนี้จะถูกนำลงไปจัดเก็บในฐานข้อมูลต่อไป

รูปที่ 3 ตัวอย่างผลลัพธ์ค่าแฮชจากอัลกอริทึม Bcrypt

ตรวจสอบความถูกต้องของค่ารหัสผ่านสามารถทำได้โดยเมื่อเริ่มต้นการทำงานจะรับค่ารหัสผ่านตั้งต้นที่ต้องการตรวจสอบยกตัวอย่างเช่น รหัสผ่านที่รับเข้ามาคือ "password" เมื่อรับเข้ามาระบบจะทำการนำค่าซอลท์ของรหัสผ่านที่จัดเก็บไว้ในฐานข้อมูลเพื่อนำมาต่อท้ายกับรหัสผ่านตั้งต้นที่รับเข้ามา คือ DM<9^C%I ซึ่งก็คือซอลท์ของรหัสผ่าน "password" ที่ได้ทำการจัดเก็บตั้งแต่เริ่มต้นในกระบวนการสร้างรหัสผ่านครั้งแรก เมื่อได้ผลลัพธ์ของค่าซอลท์แล้วจะนำรหัสผ่านตั้งต้นรวมกับค่าซอลท์ของรหัสผ่านจากฐานข้อมูลผลลัพธ์ที่ได้คือ "passwordDM<9^C%I" ต่อจากนั้นจะนำค่ารหัสผ่านที่รวมกับค่าซอลท์มาทำการสุ่มหาค่าซอลท์ของรหัสผ่านด้วยกระบวนการทำงานของอัลกอริทึม Bcrypt ตามจำนวนรอบที่กำหนด คือ 10 แล้วทำการหาผลลัพธ์ค่าแฮชของรหัสผ่านตั้งต้นที่รวมกับค่าซอลท์เมื่อได้ผลลัพธ์แล้วนำมาทำการ

เปรียบเทียบค่าผลลัพธ์ที่คำนวณกับค่าผลลัพธ์ที่จัดเก็บไว้ในฐานข้อมูลว่าผลลัพธ์ของค่าแฮชที่ได้มีค่าตรงกันหรือไม่ซึ่งเมื่อทำการเปรียบเทียบแล้วถ้าผลลัพธ์มีค่าที่ตรงกันก็แสดงให้เห็นว่ารหัสผ่านที่รับเข้ามาคือ ค่าเดียวกันกับค่าที่จัดเก็บไว้ แต่ถ้าค่าไม่ตรงกันก็แสดงให้เห็นว่าค่าของรหัสผ่านตั้งต้นที่รับเข้ามาไม่ใช่ค่าเดียวกันที่ถูกจัดเก็บไว้นั่นเอง

ผลการดำเนินการวิจัย

งานวิจัยนี้ได้ทำการสร้างความปลอดภัยรหัสผ่านด้วยเทคนิควิธีการแทรกซอลท์ร่วมกับแฮชฟังก์ชันโดยใช้อัลกอริทึม Bcrypt ผู้วิจัยได้ทดสอบโดยแบ่งออกเป็น 2 ด้าน ได้แก่ ด้านความเร็วในการทำงานและด้านความทนทานต่อการโจมตีดังนี้

1. **ผลทดสอบด้านระยะเวลาในการทำงาน**

การทดสอบด้านความทนทานต่อการโจมตีแล้วผู้วิจัยได้ทำการทดสอบด้านความเร็วในการทำงานโดยทำการทดสอบโดยการสร้างและจัดเก็บรหัสผ่านด้วยวิธีการที่นิยมใช้กันปัจจุบัน [14] จัดเก็บลงในฐานข้อมูล ซึ่งข้อมูลของรหัสผ่านที่นำมาใช้ในการทดสอบเป็นข้อมูลที่ได้จากการเก็บรวบรวมว่าเป็นรหัสผ่านที่มีผู้ใช้นิยมใช้มากซึ่งประกอบด้วยรหัสผ่านทั้งหมด 10,000 รหัสผ่าน [15] ผลที่ได้จากการทดสอบพบว่าวิธีการที่ผู้วิจัยได้นำเสนอใช้เวลา 2.59 มิลลิวินาทีต่อรหัสผ่าน ซึ่งช้ากว่าวิธีการแบบไม่มีซอลท์ที่ใช้เวลาเพียง 2.01 มิลลิวินาที และแบบ Double md5 ที่ใช้เวลาเพียง 2.04 และแบบ Fix salt ที่ใช้เวลา 2.11 มิลลิวินาที และใช้เวลาใกล้เคียงกับวิธีการแบบ Bcrypt ปกติซึ่งมีขนาดความยาวของผลลัพธ์เท่ากันที่ใช้เวลา 2.24 มิลลิวินาที ดังตารางที่ 2

ตารางที่ 2 เวลาที่ใช้ในการสร้างรหัสผ่านในกระบวนการต่าง ๆ

รูปแบบวิธี	เวลา (มิลลิวินาที)
No salt 128 bit (md5($password))	2.01
Double md5 128 bit (md5(md5($password)))	2.04
Fix salt 128 bit (md5($password.fixsalt))	2.11
Bcrypt 184 bit (bcrypt($passowd))	2.24
Bcrypt * 184 bit (bcrypt($password+salt))	2.59

* วิธีที่นำเสนอ

2. **ผลการทดสอบด้านความทนทานต่อการโจมตี**

ทำการทดสอบโดยจำลองสถานการณ์การโจมตีโดยทำการทดสอบด้านความปลอดภัยต่อการโจมตี ซึ่งรหัสผ่านที่นำทดสอบในงานวิจัยนี้จะใช้รายงานรหัสผ่านที่ทำการรวบรวมโดย SplashData [15] ซึ่งเป็นรหัสผ่านที่แย่สุดจากรายงานในปี 2016 จำนวน 25 รหัส ดังรูปที่ 4 โดยการนำรหัสผ่านทั้งหมดมาทำการจัดเก็บรหัสผ่านโดยกระบวนการแฮชฟังก์ชันด้วยอัลกอริทึม MD5 ในรูปแบบต่าง ๆ ที่กำหนดและทำการทดสอบการจัดเก็บรหัสผ่านด้วยอัลกอริทึม Bcrypt ที่ไม่ได้ทำการแทรกซอลท์และทดสอบกับรูปแบบที่ผู้วิจัยนำเสนอคือการแทรกซอลท์ร่วมกับแฮชฟังก์ชันโดยใช้อัลกอริทึม Bcrypt เช่นเดียวกัน

RANK	PASSWORD		
1	123456	14	abc123
2	password	15	admin
3	12345	16	121212
4	12345678	17	flower
5	football	18	password
6	qwerty	19	dragon
7	1234567890	20	sunshine
8	1234567	21	master
9	princess	22	hottie
10	1234	23	loveme
11	login	24	zaq1zaq1
12	welcome	25	password1
13	solo		

รูปที่ 4 ข้อมูลรหัสผ่านที่แย่ซึ่งถูกใช้ในปี 2016

หลังจากนั้นทำการทดสอบการโจมตีรหัสผ่าน ด้วยโปรแกรม Hashcat ซึ่งเป็นโปรแกรมที่ใช้ในการโจมตีค่าแฮชโดยใช้รูปแบบคำสั่งของการโจมตี [16] ได้ดังนี้

hascat -m [Hash-type] -a [Attack-mode] hashfile

ยกตัวอย่างเช่น

hashcat -m 3200 -a 3 bcrypt.txt

ผลการทดสอบพบว่าวิธีการจัดเก็บรหัสผ่านด้วยอัลกอริทึม MD5 ด้วยรูปแบบวิธีทั่วไป ซึ่งสามารถทำการโจมตีรหัสผ่านได้ 24 รหัส จากทั้งหมด 25 รหัส คิดเป็น 96 % จากรหัสผ่านทั้งหมด นอกจากนั้นผู้วิจัยได้ทำการทดสอบการโจมตีจากการจัดเก็บรหัสผ่านด้วยอัลกอริทึม Bcrypt โดยที่ไม่ได้ทำการแทรกซอลท์ลงไปในรหัสผ่าน ผลจากการทดสอบพบว่าไม่สามารถทำการโจมตีได้ นอกจากนี้รูปแบบที่ผู้วิจัยนำเสนอ คือ การสร้างความปลอดภัยรหัสผ่านโดยวิธีการแทรกซอลท์ร่วมกับแฮชฟังก์ชันโดยใช้อัลกอริทึม Bcrypt ผลจากการทดสอบพบว่าไม่สามารถโจมตีได้เช่นกันแสดงดังตารางที่ 3

ตารางที่ 3 ผลการโจมตีรหัสผ่านด้วยโปรแกรม Hashcat

วิธีการจัดเก็บรหัสผ่าน	รูปแบบ	ผลการโจมตี
No salt	md5($password)	96 %
Double md5	md5(md5($password))	96 %
Fix salt	md5($password.fixsalt)	96 %
Bcrypt	bcrypt($password)	0 %
Bcrypt *	bcrypt($password+salt)	0 %

* วิธีที่นำเสนอ

จากผลการทดสอบด้านความทนทานการโจมตีดังตารางที่ 3 แสดงให้เห็นว่าการสร้างความปลอดภัยรหัสผ่านด้วยเทคนิควิธีการแทรกซอลท์ร่วมกับแฮชฟังก์ชันโดยใช้อัลกอริทึม Bcrypt ที่ผู้วิจัยนำเสนอยังคงสามารถรักษาความลับของข้อมูลรหัสผ่านในสถานการณ์จำลองกรณีที่ระบบถูกโจรกรรมฐานข้อมูลรหัสผ่านออกไปได้ ซึ่งทำให้ผลที่ได้มีความสอดคล้องกับงานวิจัยที่ได้นำอัลกอริทึม Bcrypt เข้ามาช่วยในการรักษาความปลอดภัยข้อมูลของผู้ใช้ในการใช้งานบริการซื้อของออนไลน์ก่อนทำการบันทึกลงฐานข้อมูลโดยทำการเข้ารหัสขนาด 512 บิต [17]

สรุปผลการดำเนินงาน

งานวิจัยนี้ได้นำเสนอการสร้างความปลอดภัยรหัสผ่านด้วยเทคนิควิธีการแทรกซอลท์ร่วมกับแฮชฟังก์ชันโดยใช้อัลกอริทึม Bcrypt โดยได้ทำการจำลองสถานการณ์การโจมตีในกรณีที่ระบบถูกโจรกรรมรหัสผ่านที่จัดเก็บในฐานข้อมูลออกไป ผลจากการทดสอบพบว่าถ้าหากต้องการรักษาความปลอดภัยของรหัสผ่านวิธีการเพิ่มซอลท์ร่วมกับรหัสผ่านมีความจำเป็นหากใช้แฮชฟังก์ชันทั่วไปในการจัดเก็บรหัสผ่าน เช่น อัลกอริทึม MD5 แต่ในงานวิจัยนี้ได้ทำการทดลองนำแฮชฟังก์ชันจำพวก Slow Hash มาทำการทดสอบสองรูปแบบด้วยกัน คือ อัลกอริทึม Bcrypt อย่างเดียวในการจัดเก็บรหัสผ่าน ซึ่งทำให้สามารถที่จะลดความเสี่ยงที่เกิดจากการโจมตีด้วยโปรแกรม Hashcat ได้ เนื่องจากอัลกอริทึม Bcrypt มีจำนวนรอบการทำงานตั้งแต่ 4 รอบ ไปจนถึง 31 รอบการทำงาน ซึ่งส่งผลให้การประมวลผลของการสร้างค่าแฮชของผลลัพธ์มีความช้าหรือเร็วนั้นขึ้นอยู่กับความเร็วของการประมวลผลของซีพียูที่ใช้ในการประมวลผลและยิ่งถ้าหากใช้วิธีการแทรกซอลท์ร่วมกับแฮชฟังก์ชันโดยใช้อัลกอริทึม Bcrypt ที่ผู้วิจัยนำเสนอก็ยังคงไม่สามารถโจมตีได้ซึ่งทำให้สามารถรักษาความลับของรหัสผ่านเอาไว้ได้เช่นกัน ถึงแม้ว่าระยะเวลาในการสร้างรหัสผ่านที่ใช้รูปแบบอัลกอริทึม Bcrypt ที่ผู้วิจัยนำเสนอ คือ การเพิ่มซอลท์เข้าไปจะมีระยะเวลาที่มากกว่ารูปแบบ Bcrypt เพียงอย่างเดียว แต่การแทรกซอลท์เพิ่มเติมจะช่วยทำให้มีความซับซ้อนเพิ่มมากขึ้น ซึ่งมีผลทำให้ระยะเวลาในการประมวลที่ช้ากว่ารูปแบบของอัลกอริทึม Bcrypt เพียงอย่างเดียว ซึ่งจะทำให้การโจมตีสามารถทำได้ช้าลงเช่นเดียวกัน ทำให้รูปแบบอัลกอริทึม Bcrypt แบบเพิ่มซอลท์มีความปลอดภัยเพิ่มมากยิ่งขึ้น แต่อย่างไรก็ตามในอนาคตสามารถทำการทดสอบการโจมตีได้ด้วยรูปแบบวิธีอื่น หรือทำการทดสอบด้วยโปรแกรมอื่นที่ใช้ในการโจมตี

References

[1] Stallings, W. (2005). **Cryptography and Network Security**. Prentice Hall

[2] Kumari, C. S. and Rani, M. D. (2013). Hacking Resistance Protocol for Securing Passwords using Personal Device. In **7th International Conference on Intelligent Systems and Control (ISCO).** pp. 458-463

[3] Kumar, H. (2013). Rainbow Table to Crack Password using MD5 Hashing Algorithm. In **IEEE Conference on Information and Communication Technologies**.

[4] Li, P., Sui, Y., and Yang, H. (2010). The Parallel Computation in One-Way Hash Function Designing. In **International Conference on Computer, Mecharronics, Control and eLectornic Enginering (CMCE)**. pp. 189-192

[5] Provos, N. and Mazeries, D. (1999). A Future Adaptable Password Scheme. In **Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference, June 6-11, 1999**. Monterey, California, USA 1999.

[6] Cindy, M., Kioon, A., Wang Z., and Deb, S. (2013). Security Analysis of MD5 algorithm in Password Storage. In **Proceeding of the 2nd International Symposium on Computer, Communication, Control and Automation, (ISCCCA-13)**. DOI: 10.2991/isccca.2013.177.

[7] Boonkrong, S. (2012). Security of Passwords. **Journal of Information Technology**. Vol. 8, No. 2, pp. 112-117

[8] Sood, S. K. Sarje, A. K., and Singh, K. (2009). Cryptanalysis of Password Authentication Schemes: Current Status and Key Issues. In **Methods and Models in Computer Science, 2009. ICM2CS 2009. Proceeding of International Conference on**, Dec 2009. pp. 1-7

[9] Ma, W., Campbell, J., Tran, D., and Kleeman, D. (2010). Password Entropy and Password Quality. In **4th International Conference on Network and System Security (NSS)**. pp. 583-587

[10] Zheng, X. and Jin, J. (2012). Research for the Application and Safety of MD5 Algorithm in Password Authentication. In **Proceedings of the 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)**. pp. 2216-2219

[11] Gauravaram, P. (2012). Security Analysis of salt||password Hashes. In **Advanced Computer Science Applications and Technologies (ACSAT), 2012 International Conference on**. pp. 25-30

[12] Kumar, H., Kumar, S., Joseph, R., Kumar, D., Shrinarayan Singh, S. K., Kumar, P., and Kumar, H. (2013). Rainbow Table to Crack Password using MD5 Hashing Algorithm. In **IEEE Conference on Information Communication Technologies (ICT)**. pp. 433-439

[13] Boonkrong, S. and Somboonpattanakit, C. (2016). Dynamic Salt Generation and Placement for Secure Password Storing. **International Journal of Computer Science**. Vol. 43, No. 1, pp. 27-36

[14] Whitney, L. (2017). **Millions of LinkedIn passwords reportedly leaked online**. Access (4 July 2017). Available (https://www.cnet.com/news/millions-of-linkedin-passwords-reportedly-leaked-online/)

[15] Slain, M. (2016). **Worst Passwords of 2016**. Access (14 March 2017). Available (https://www.teamsid.com/worst-passwords-2016)

[16] University of South Wales: Information Security & Privacy. (2013). **Kali How to crack passwords using Hashcat - The Visual Guide**. Access (14 March 2017). Available (http://uwnthesis.wordpress.com/2013/08/07/kali-how-to-crack-passwords-using-hashcat/)

[17] Sriramya, P. and Karthika, R. A. (2015). Providing Password Security By Salted Password Hashing Using Bcrypt Algorithm. **ARPN Journal of Engineering and Applied Sciences**. Vol. 10, Issue 13, pp. 5551-5556